IN THE CLAIMS

Claim 1 (Currently Amended): A process for providing a run-time representation of specified characteristics of a previously developed object-oriented software program, the software program including a plurality of object classes and further including object related methods belonging to respective object classes, the process comprising:

sensing at least one complex method call by examining the source code of the software program, a plurality of the methods being associated with each of the at least one complex method call, wherein the at least one complex method call includes at least one single method call;

extracting the at least one single method call from each of the at least one complex method call, wherein the extracting comprises recursively replacing the at least one single method call with a phase variable;

parsing the phase variable to resolve any polymorphic behavior associated with the extracted at least one single method call;

generating a set of information for each of the methods from the at least one single method call, the information set for a particular method containing at least a name of the particular method and the run-time object class of the plurality of object classes to which the particular method belongs, wherein at least one of the object related methods in the software program is a polymorphic method and the run-time object class is determined based on the parsing of the phase variable; and

constructing a representation of interactions between objects of the software program from the information contained in the method information sets.

Claim 2 (Currently Amended): The process of claim 1, wherein the at least one single method call comprises a method parameter of the at least one complex method call or a continuous method call.

Claim 3 (Previously Presented): The process of claim 1, wherein the process further comprises extracting the name and class of each of the methods from the software program.

Claim 4 (Previously Presented): The process of claim 1, wherein the extracting comprises output a given complex method call as multiple lines, wherein at least one of the multiple lines contains the at least one single method call.

Claim 5 (Previously Presented): The process of claim 4, wherein the extracting comprises:

a first parsing phase configured to separate any casting operations included in the given complex method call;

a second parsing phase configured to isolate any method parameters included in the given complex method call; and

a third parsing phase configured to resolve any continuous method calls included in the given complex method call into multiple lines, each containing one of the at least one single method call.

Claim 6 (Previously Presented): The process of claim 5, wherein the first parsing phase is implemented prior to the second parsing phase, and the second parsing phase is implemented prior to the third parsing phase.

Claim 7 (Previously Presented): The process of claim 6, wherein the generating includes parsing an output provided by the third parsing phase to determine the correct object class for each of the methods.

Claim 8 (Previously Presented): The process of claim 7, wherein the process further comprises determining whether a method is a user-defined method or a standard application programming interface method.

Claim 9 (Previously Presented): The process of claim 1, wherein the constructing comprises constructing a sequence diagram depicting the interactions between respective objects of the software program.

Claim 10 (Previously Presented): The process of claim 9, wherein the sequence diagram displays a condition of a method call to indicate that the method call occurs only when the condition is evaluated to be true.

Claim 11 (Previously Presented): The process of claim 1, wherein the software program is in the form of source code.

Claim 12 (Previously Presented): The process of claim 1, wherein the software program is written in JAVA™ software code.

Claim 13 (Previously Presented): The process of claim 1, wherein the software program is written in C++ software code.

Claim 14 (Canceled).

Claim 15 (Previously Presented): The process of claim 1, wherein at least one of the object related methods in the software program is related to an inheritance feature, and the extraction includes tracking an inheritance path until it reaches a parent object class to which the method is defined.

Claim 16 (Currently Amended): A computer program product in a computer readable media for providing a run-time representation of specified characteristics of a previously developed object-oriented software program, the software program including a plurality of object classes and object related single methods belonging to respective object classes, the software program further including at least one complex method call containing a plurality of the single methods, the computer program product comprising:

first instructions for extracting a plurality of individual method calls from each of the at least one complex method call by examining the source code of the software program, wherein the first instructions comprise recursively replacing each of the single methods with a phase variable;

second instructions for parsing the phase variable to resolve any polymorphic behavior associated with the extracted plurality of individual method calls;

third ~~second~~ instructions for storing in a data base a set of information for each of the single methods, the set of information for a particular single method containing at least a name of the particular method and the run-time object class of the plurality of object classes to which the particular method belongs, wherein at least one of the object related single methods in the software program is a polymorphic method and the run-time object class is determined based on the parsing of the phase variable; and

fourth ~~third~~ instructions for graphically constructing and graphically outputting a representation of interactions between objects of the software program from the information contained in the set of information.

Claim 17 (Currently Amended): The computer program product of claim 16, wherein further comprising fifth ~~fourth~~ instructions for extracting the name and the object class of each of the single methods from the software program.

Claim 18 (Currently Amended): The computer program product of claim 17, wherein the first instructions comprise sixth ~~fifth~~ instructions for outputting a given complex method callas multiple lines, each containing one of the single method calls.

Claim 19 (Previously Presented): The computer program product of claim 18, wherein the first instructions sequentially implement a first parsing phase to separate any casting operations included in the given complex method call, a second parsing phase to isolate any method parameters included in the given complex method call, and a third parsing phase to

resolve the given complex method call into multiple lines, each containing one of the single method calls.

Claim 20 (Previously Presented): The computer program product of claim 19, wherein the third instructions constructs a sequence diagram depicting the interactions between respective objects of the software program.

Claim 21 (Previously Presented): The computer program product of claim 20, wherein the software program is in the form of source code.

Claim 22 (Currently Amended): An apparatus for providing a sequence diagram representing specified run-time characteristics of a previously developed object-oriented software program, the software program including a plurality of object classes and further including object related methods belonging to respective object classes, the apparatus comprising:

a computer system having a display;

means for sensing at least one complex method call by examining the source code of the software program, a plurality of the methods being associated with each of the at least one complex method call;

Method Detail Parser means for extracting a plurality of single method calls from each of the at least one complex method call, the Method Detail Parser means comprises recursively replacing the associated methods with a phase variable;

means for parsing the phase variable to resolve any polymorphic behavior associated with the extracted plurality of single method calls;

means for generating a set of information for each of the object related methods from the single method calls, the set of information for a particular object related method containing at least a name of the particular method and the run-time object class of the plurality of object

classes to which the particular method belongs, wherein at least one of the object related methods in the software program is a polymorphic method and the run-time object class is determined based on the parsing of the phase variable; and

means for constructing a sequence diagram representing interactions between objects of the software program from the information contained in the sets of information and outputting the sequence diagram on the display.

Claim 23 (Previously Presented): The apparatus of claim 22, wherein the apparatus further comprises Method Information Parser means for extracting the name and the object class of each of the methods from the software program.

Claim 24 (Previously Presented): The apparatus of claim 23, wherein the Method Detail Parser means is operable to output a given complex method callas multiple lines, each containing one of the single method calls.

Claim 25 (Previously Presented): The apparatus of claim 24, wherein the Method Detail Parser means is configured to separate any casting operations included in the given complex method call during a first parsing phase, to isolate any method parameters included therein during a second parsing phase, and resolve the given complex method call into multiple lines, each containing one of the single method calls, during a third parsing phase.

Claim 26 (Previously Presented): The apparatus of claim 25, wherein the first parsing phase is implemented prior to the second parsing phase, and the second parsing phase is implemented prior to the third parsing phase.

Claim 27 (Previously Presented): The apparatus of claim 26, wherein the constructing means comprises a drawing engine for depicting interactions between respective objects of the software program.

Claim 28 (Previously Presented): The apparatus of claim 27, wherein the software program is in the form of source code.